FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Performance Measurements of BLACS Routines on CRAY T3E

Tobias Oechtering, Inge Gutheil

FZJ-ZAM-IB-2000-06

Juli 2000

(letzte Änderung: 04.07.2000)

Performance Measurements of BLACS Routines on CRAY T3E

Tobias Oechtering and Inge Gutheil

John von Neumann-Institut für Computing
Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich GmbH

July 4, 2000

Abstract

The ScaLAPACK [1] library is based on the BLACS (Basic Linear Algebra Communication Subroutines) [2] library. Unfortunately the optimized SHMEM-based CRAY T3E BLACS library contained in libcomm.a has a bug which leads to problems when sub-grids are created. Therefore it is sometimes necessary to use a MPI-based public domain BLACS library by linking the libraries libblacs.a and liblacsF77init.a.

In this report performance measurements are presented using BLACS routines of both libraries in order to gain more information about the differences. Communication routines and a global combine operation will be compared.

1 Performance of BLACS

All measurements were done on CRAY T3E-1200 at Research Centre Jülich during March 2000. On CRAY T3E there is always one MPI process per processor, thus process and processor are synonyms here.

1.1 Communication Routines

1.1.1 Point to Point Communication

Send/Receive a general matrix from one process to another using:

- **_GESD2D(ICTXT,M,N,A,LDA,RDEST,CDEST)**
- **_GERV2D(ICTXT,M,N,A,LDA,RSRC,CSRC)**

where _ can be I for type INTEGER, S for type Single precision REAL (64 bit on CRAY T3E), or C for Complex (2 times 64 bit).
For measurement of the elapsed time for sending an integer matrix the following Fortran lines are used:

```
   ...
CALL MPI_BARRIER(MPI_COMM_WORLD, ierr)
IF ( IAM.EQ.0 ) THEN
   elp1=MPI_wtime()
ENDIF
DO I=1,LOOPS
   IF ( IAM.EQ.0 ) THEN
      CALL IGESD2D(CONTXT,M,N,A,DIM,0,1)
   ENDIF
   IF ( IAM.EQ.1 ) THEN
      CALL IGERV2D(CONTXT,M,N,A,DIM,0,0)
   ENDIF
   IF ( IAM.EQ.1 ) THEN
      CALL IGESD2D(CONTXT,M,N,A,DIM,0,0)
   ENDIF
   IF ( IAM.EQ.0 ) THEN
      CALL IGERV2D(CONTXT,M,N,A,DIM,0,1)
   ENDIF
ENDDO
CALL MPI_BARRIER(MPI_COMM_WORLD, ierr)
IF ( IAM.EQ.0 ) THEN
   elp2=MPI_wtime()
   elpt=(elp2-elp1)/(2*LOOPS)
ENDIF
   ...
```

A matrix A possessing a number of 0, 1, 100, 10000 or 1000000 entries of type integer or real is transmitted from process zero to process one and reverse. This has to be done, as transmitting only in one direction exceeds the number of messages that can be queued on one process. Measuring just one transmission does not give reproducible execution times due to the strong dependence

on the load of the used communication network and the very short execution time for one single send/receive of a short message. After 1000 iterations the additional time of barrier execution is negligible and an average of execution time is achieved. The measured time is written in *elpt*.

Results As *Table 1* shows there are just small differences in transmitting real or integer matrices. That is reasonable because integer and real both need 64 bits of memory on CRAY T3E [4]. The table is listing the elapsed time using the Cray optimized library, but there are also just small differences when using the public domain library.

number of matrix-entries	of type INTEGER	of type REAL
0	2.89916E-7	2.67267E-7
1	1.15469E-5	1.17330E-5
100	1.91980E-5	1.91645E-5
10000	5.82272E-4	5.81373E-4
1000000	5.57217E-2	5.55031E-2

Table 1: Point to Point Communication, execution times in sec - INTEGER vs. REAL using Cray BLACS

Comparing the performance of point to point communication using the Cray optimized and public domain library shows a speedup of the Cray optimized implementation.

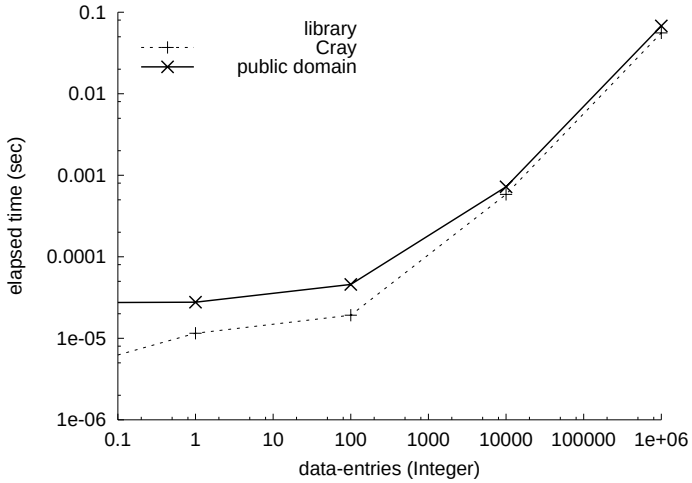

Figure 1: Point to Point Communication - performance

Between 100 and 10000 data entries the internal MPI protocol [3] is switched from short to eager protocol. Short protocol buffers the message envelope and

the message data in a pre-allocated slot at the receiver, eager protocol buffers envelope at the receiver and the message data is buffered in a temporarily allocated buffer at the receiver or sender. Speedup decreases from 2.4 to 1.2 switching the protocol. Cray implemented an extra treatment transmitting an empty message while the public domain library uses standard protocol for small messages. This leads to a speedup of 90.2 for empty messages.

data-entries:	0	1	100	10000	1000000
speedup:	90.2	2.40	2.39	1.24	1.22

Table 2: Point to Point Communication - speedup of Cray BLACS compared to public domain BLACS

1.1.2 Broadcast Communication

Send a general matrix to all processes or subsection of processes in scope using:

- **_GEBS2D(ICTXT,SCOPE,TOP,M,N,A,LDA)**
- **_GEBR2D(ICTXT,SCOPE,TOP,M,N,A,LDA,RSRC,CSRC)**

where _ again can be I, S, or C.
For measurement of the elapsed time for broadcasting an integer matrix the following Fortran lines are used:

```
   ...
DO I=0,LOOPS
   CALL BLACS_BARRIER(CONTXT,'Row')
   elp1=MPI_wtime()
   IF ( IAM.EQ.0 ) THEN
      CALL IGEBS2D(CONTXT,'Row','',M,N,A,DIM)
   ELSE
      CALL IGEBR2D(CONTXT,'Row','',M,N,A,DIM,0,0)
   ENDIF
   elp2=MPI_wtime()
   elpt=(elp2-elp1)
   CALL SGAMX2D(CONTXT,'All','',1,1,elpt,1,X,Y,-1,0,0)
ENDDO
   ...
```

A matrix A possessing a number of 0, 1, 100, 10000 or 1000000 entries is broadcast from process zero to a variable number of processes. The maximum of the elapsed times of all processes is reported. After five iterations the measured time is reproducible. The result is written in *elpt*. The default topology TOP=' ' is used meaning that the MPI broadcast topology will be used. Only the times for broadcasting a matrix of integers are measured as type real should deliver the same results.

Results Plotting elapsed time vs. number of receiving processes for different package-sizes to be transmitted shows the propagation.

In *Figure 3* it is clearly cognizable that the propagation proceeds like an avalanche. At 1, 3, 7 and 15 processes steps can be observed. In the first step one process sends and one process receives data. In the second step those two processes send to two other processes - already three processes received data. In the following steps all processes which have already received data propagate it to others until all supposed processes received data.

So the elapsed time broadcasting data depends on the number of steps which are necessary to propagate. Between those steps the elapsed time remains almost constant.

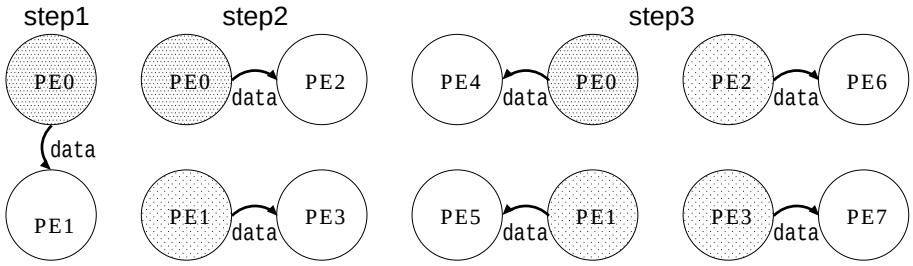

Figure 2: Broadcast Communication - data propagation

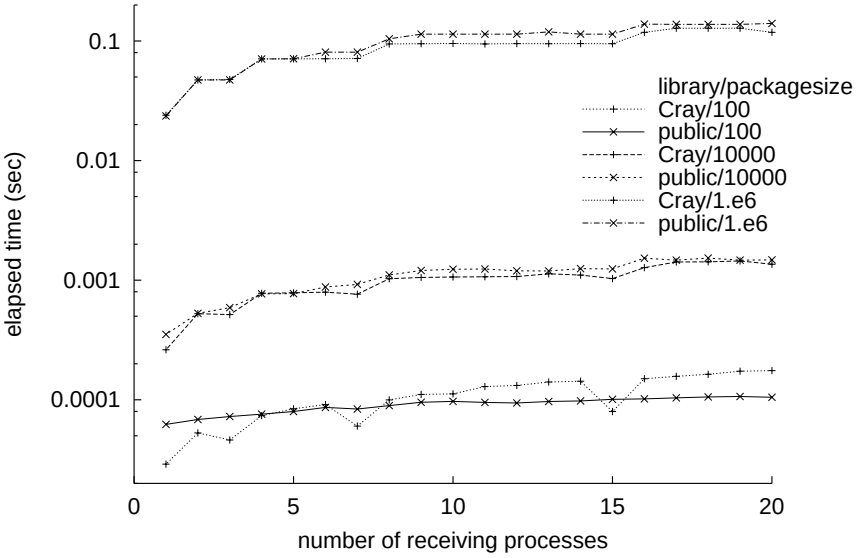

Figure 3: Broadcast Communication - tree structure

In *Figure 3* the elapsed time axis is scaled logarithmically and the transmission of 1 and no entries are not plotted, so be careful comparing time differences

between both libraries. The Cray implementation seems to be faster in special cases, broadcasting to 1, 3, 5 or 15 processes. This advantage gets lost transmitting larger data-packages.

Table 3 lists elapsed time vs. data-entries using both libraries. The time measurements broadcasting small data-packages are strongly influenced by communication load of other processes on the machine.

data-entries type INTEGER	elapsed time (in sec) Cray lib	elapsed time (in sec) public domain lib	speedup
0	3.31401E-5	6.36577E-5	1.82
1	9.98973E-5	8.64267E-5	0.86
100	1.12056E-4	9.70363E-5	0.86
10000	1.06275E-3	1.23453E-3	1.16
1000000	9.5279E-2	0.11404	1.20

Table 3: Broadcast Communication - data from 1 $\longrightarrow$ 10 processes

The examination case, broadcasting no data, performs better using Cray lib. Small data-packages are faster distributed using public domain lib. When data-packages are getting larger, a faster transmission using Cray implementation leads to speedups of up to 20%. These trends are similar for varying numbers of receiving processes.

1.2 Global Combine Operations

1.2.1 Maximum

Perform element-wise $|MAX|$ operation on general matrices using:

- **_GAMX2D(ICTXT,SCOPE,TOP,M,N,A,LDA,RA,CA,RCFLAG, RDEST,CDEST)**

where _ again can be I, S, or C. Element-wise indicates that each element of the input matrix will be compared to the corresponding element from all other processes' matrix to find the maximum.
For measurement of the elapsed time for finding the element-wise maximum of real general matrices the following Fortran lines are used:

```
   ...
DO I=1,LOOPS
    A=C
    elp1=MPI_wtime()
    CALL SGAMX2D(CONTXT,'All','',M_A,N_A,A,LDA,RA,CA,-1,-1,-1)
    elp2=MPI_wtime()
    elpt=(elp2-elp1)
    CALL SGAMX2D(CONTXT,'All','',1,1,elpt,1,X,Y,-1,0,0)
ENDDO
   ...
```

The matrix C assigns random values to the matrix A to be compared. After the execution of the routine data entries in A may be overwritten. Matrix A

possesses different sizes (10x10...500x500) for different workloads. The maximum of the elapsed times over all processes is reported. After five iterations the measured time is reproducible and written in *elpt*. Again only the default topology and now matrices of type real are examined.

Results *Table 4* shows the performance of the Combine Function $|MAX|$ varying the matrix-size to be compared and the arrangement of the 16 processes using Cray optimized and public domain library. The elapsed times given in *Table 4* are the results for computing the $|MAX|$ and storing the location of that maximum.

matrix-size	elapsed time (in sec) using Cray library		
type REAL	1 x 16 grid	2 x 8 grid	4 x 4 grid
10 x 10	3.64303E-4	3.67164E-4	3.78370E-4
50 x 50	3.52609E-3	3.62730E-3	3.72886E-3
100 x 100	1.30370E-2	1.38238E-2	1.41824E-2
500 x 500	0.31873	0.33497	0.34488
	elapsed time (in sec) using public domain library		
	1 x 16 grid	2 x 8 grid	4 x 4 grid
10 x 10	4.27603E-4	4.57167E-4	4.49419E-4
50 x 50	3.95023E-3	4.18961E-3	4.21440E-3
100 x 100	1.39807E-2	1.37597E-2	1.39526E-2
500 x 500	0.29858	0.31435	0.32107

Table 4: Combine Function $|MAX|$ - performance with different shapes of a 16 processor grid

Comparing both libraries shows a faster implementation of the Cray library for small, 10 x 10 and 50 x 50, matrices and then slightly faster execution of the public domain library. Advantages of the Cray library with small data entries are assumed to rely on faster startup times based on SHMEM routines. This effect could also be observed when evaluating the communication routines. The catch up of the MPI based public domain library is assumed to rely on the fact that the MPI library for CRAY T3E-1200 is also optimized and the public domain BLACS library uses the MPI global reduce operations.

Regarding the shape of the process grid increasing times with increasing number of rows in the process grid could be observed up to a square grid. Then the execution times decreased with the number of rows, so the highest execution time was reached with a square grid.

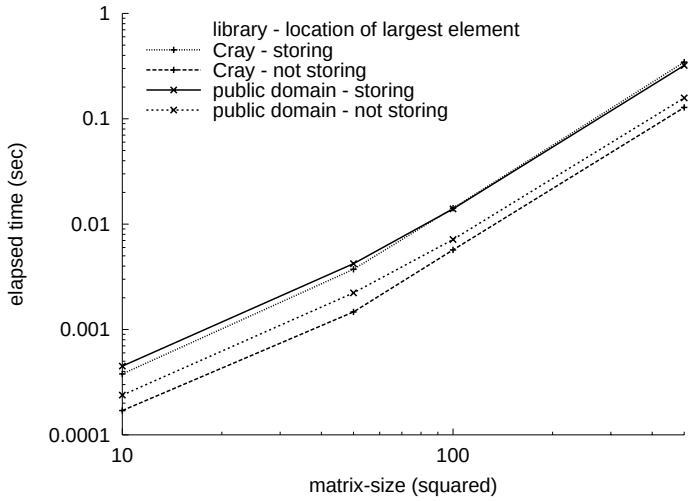

Figure 4: Combine Function $|MAX|$ - location of largest element stored or not

In the routine the variable $RCFLAG$ determines whether the row and column index of the process that provides the maximum should be stored in RA and CA or not. The graph in *Figure 4* shows the differences using both libraries. When the flag is set the location of the largest element will be stored, you cannot detect any divergences.

Not to store the locations leads to speedups. Using public domain library this halved execution time. Using Cray library gives approximately another 25% speedup.

$RDEST$ and $CDEST$ indicate the recipient of the result. If one of them is set to -1 all processes will receive the answer. Comparing the performance in *Figure 5* of both libraries with respect to one or all processes to be recipient shows unexpected behavior.

Transmitting the result to one process using the Cray library has the same performance as transmitting the result to all processes. This made more exact examination necessary. It was found that there are no differences in execution. Indicating one recipient the Cray implementation transmits the result to all processes, whereas the implementation of the public domain library transmits the result just to the indicated process and therefore achieves a speedup of about 25%.

2 Acknowledgements

During my traineeship I learned a lot about the use of massively parallel processor systems, the Cray systems at Research Centre Jülich, and about handling of some useful tools for my further studies and became sensitized to problems of parallel computing.

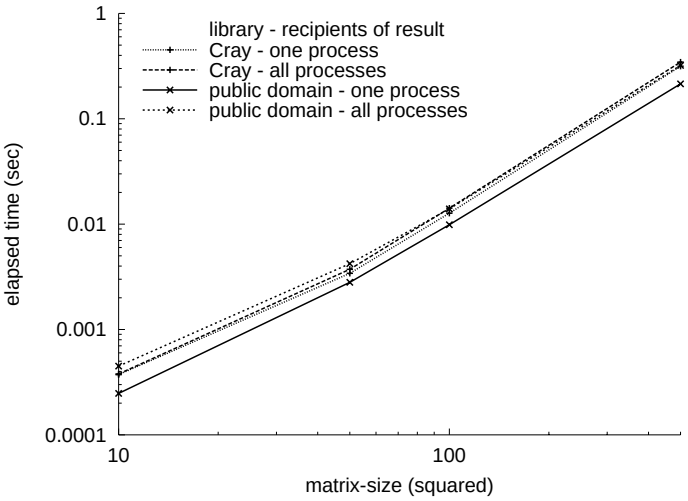

Figure 5: Combine Function $|MAX|$ - recipients of result

I want to thank Dr. Rüdiger Esser, who made this traineeship possible, and Dr. Johannes Grotendorst, who looked after me and all doctor and diploma candidates, who helped me in all situations. They all made this traineeship a rich experience.

References

[1] L. S. Blackford, J. Choi, and A. Cleary et al., *ScaLAPCK Users' Guide*, SIAM Philadelphia, 1997. Also available via www under http://www.netlib.org/scalapack/slug/scalapack_slug.html

[2] Jack J. Dongarra and R. Clint Whaley, *A User's Guide to the BLACS v1.1*, http://www.netlib.org/blacs/lawn94.ps, 1997.

[3] Rolf Rabenseifner, *Message Passing Interface (MPI), Course Material*, Rechenzentrum der Universität Stuttgart, 2000.

[4] Valentina Tikko, *The Cray Systems at Research Centre Jülich, Volume 1, 2*, Zentralinstitut für Angewandte Mathematik, 2000. FZJ-ZAM-BHB-0138, FZJ-ZAM-BHB-0139